Atrevido: SemiDynamics Out-of-Order RISC-V Core

Roger Espasa, PhD, CEO & Founder

Semidynamics

Semidynamics RISC-V Cores

AVISPADO 222

2-wide In-Order

Gazzillion Misses™

VPU 1.0

ATREVIDO 222

2-wide Out-of-Order

Gazzillion Misses™

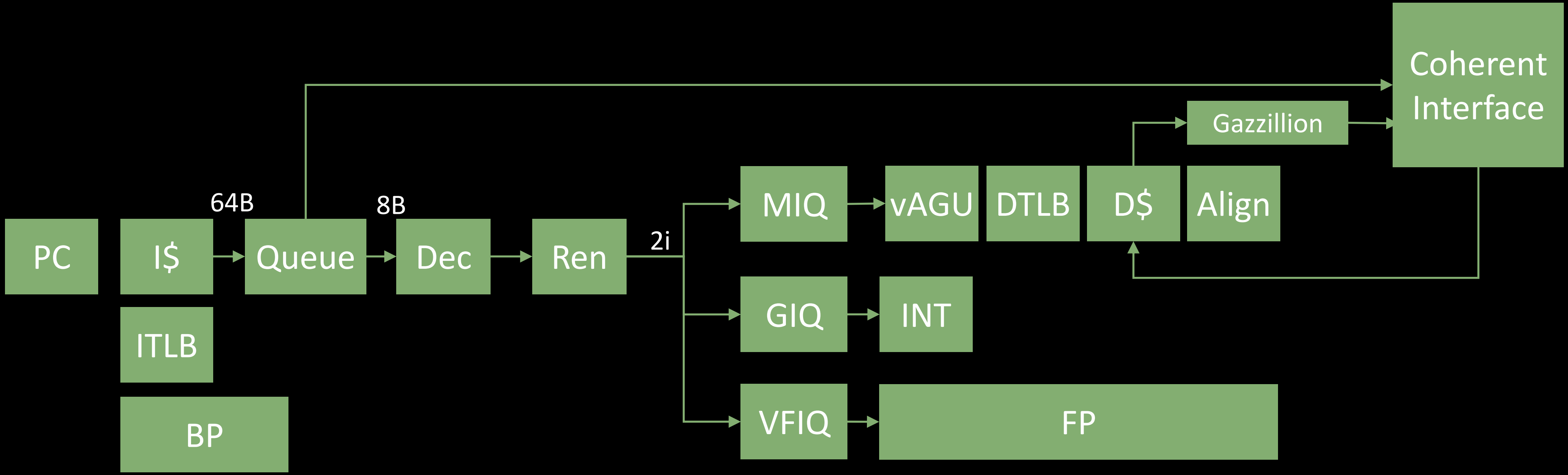
VPU 1.0

TODAY's FOCUS

IP cores available for licensing

ATREVIDO 222

RISCV64GCV



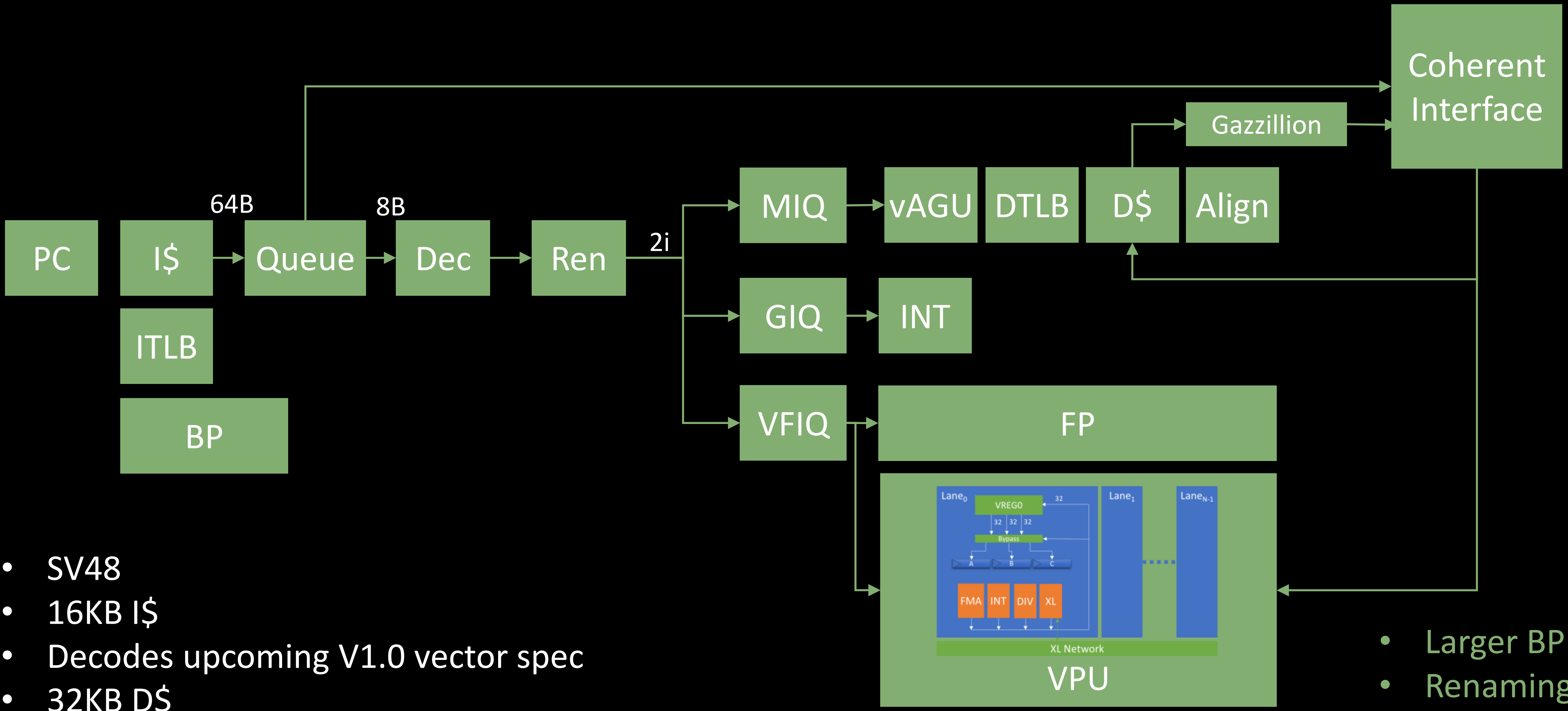
- SV48
- 16KB I$
- Decodes upcoming V1.0 vector spec
- 32KB D$
- Full hardware support for unaligned accesses
- Coherent (CHI)

- Larger BP
- Renaming
- Retirement Logic

Available for licensing

ATREVIDO 222 with OOO VPU (RVV1.0)

RISCV64GCV



- SV48
- 16KB I$
- Decodes upcoming V1.0 vector spec
- 32KB D$
- Full hardware support for unaligned accesses
- Coherent (CHI)

- Larger BP
- Renaming
- Retirement Logic

Available for licensing

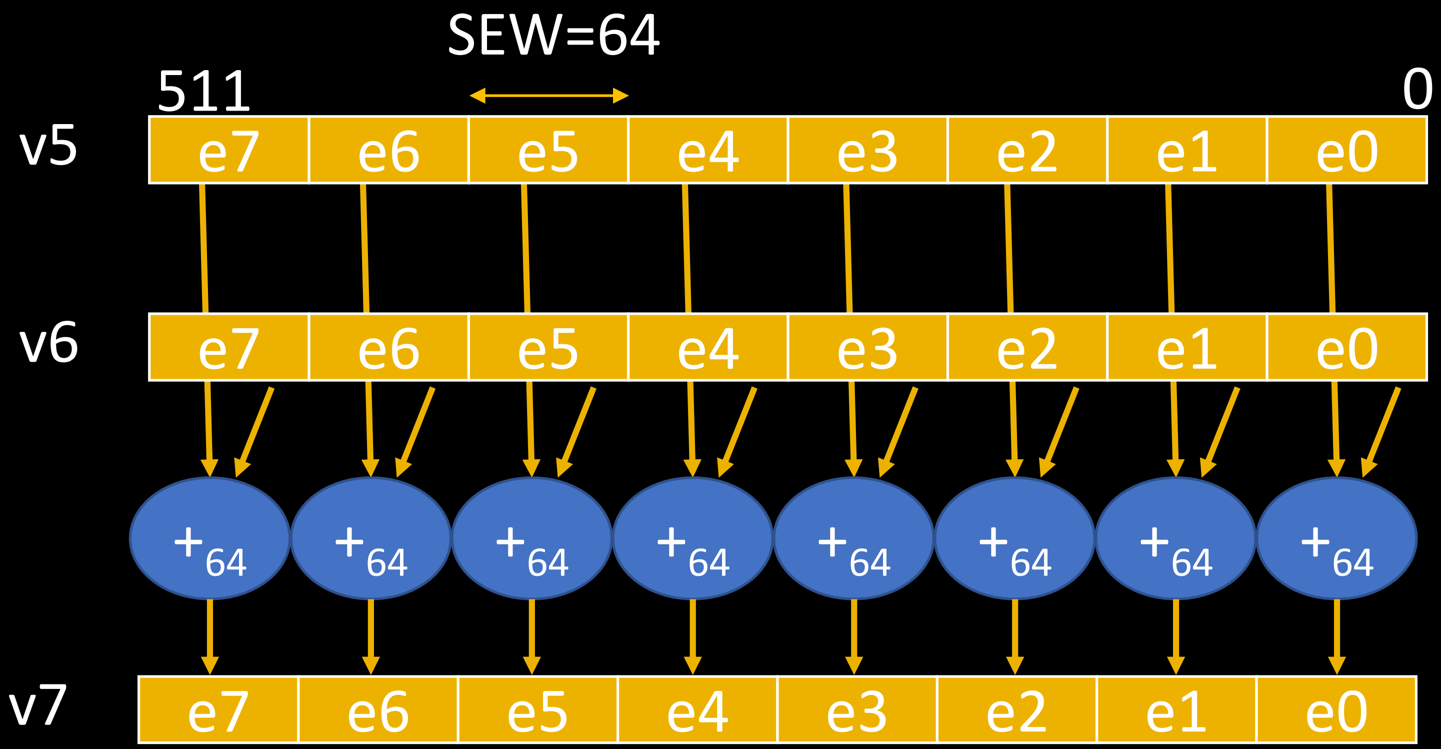
SemiDynamics' VPU

- Implements the RISC-V Vector 1.0 Specification
  - Including lmul, segmented loads
  - No vector atomics currently
- Ready for OOO support
  - Renaming
- Customizable settings
  - VLEN = vreg size = from 128b to 4096b
  - Data path width = from 128b to 512b
  - Fast cross-lane network for slide/rgather/compress/expand

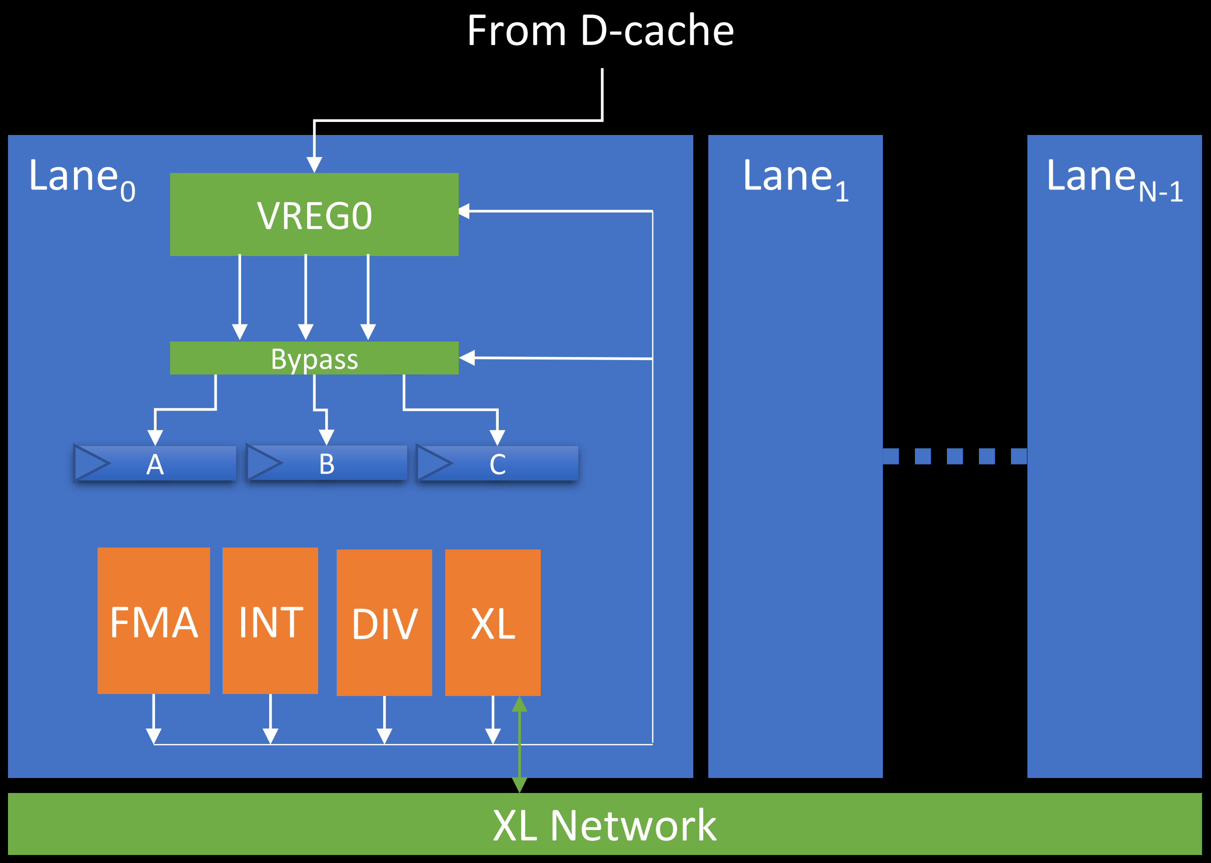
Available for licensing 3 months after V-spec freeze

One slide reminder of RVV

```
vsetvli    x9 ← x8, e64, m1
vadd.vv    v7 ← v5, v6
```



VPU Block Diagram



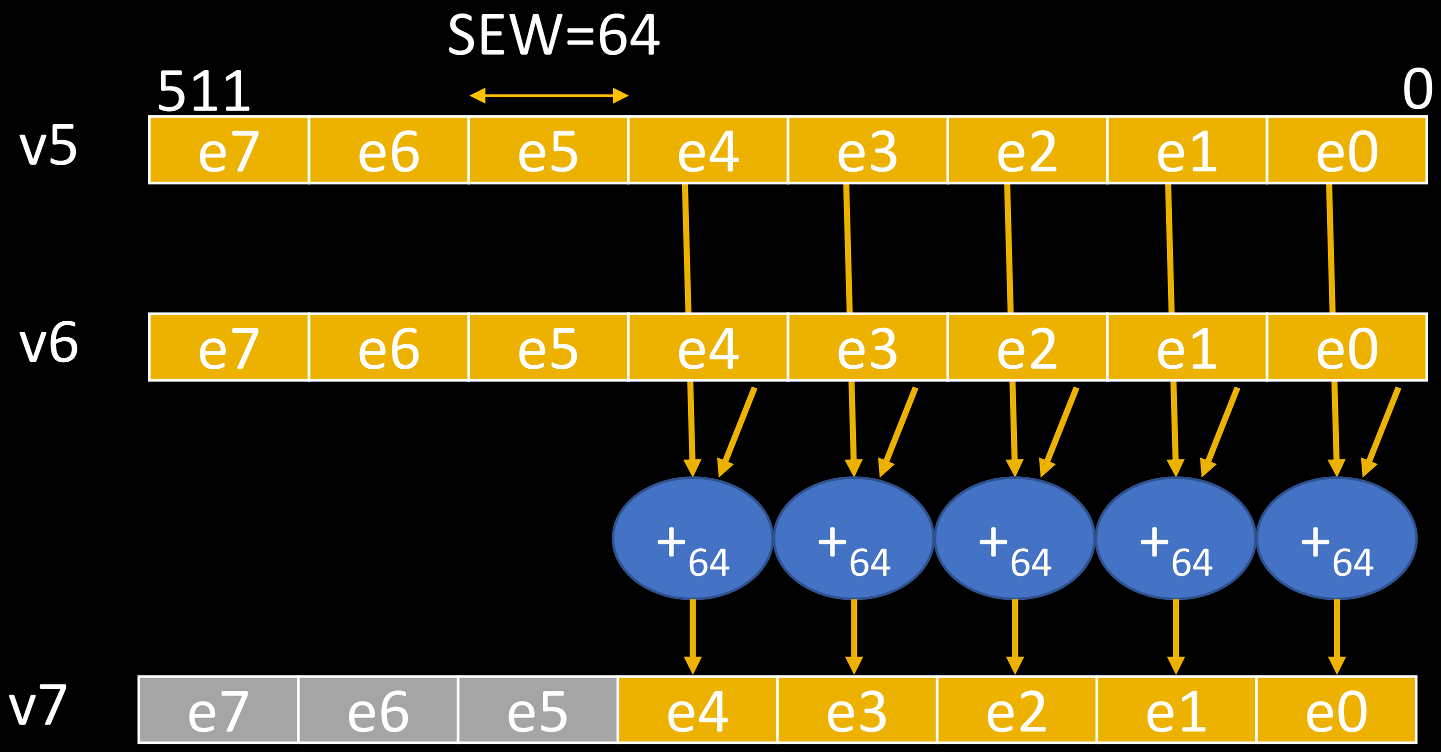
- Lane based organization
- Full cache-line bus from D-cache
- Units per lane
  - FMA
  - INT
  - DIV
  - XL: Cross-lane (rgather, …)
- Full masking support

Available for licensing 3 months after V-spec freeze

~~One~~ Two slide reminder of RVV:

```
vsetvli    x9 ← x8, e64, m1
vadd.vv    v7 ← v5, v6
```

Assume x8 = 5



Renaming

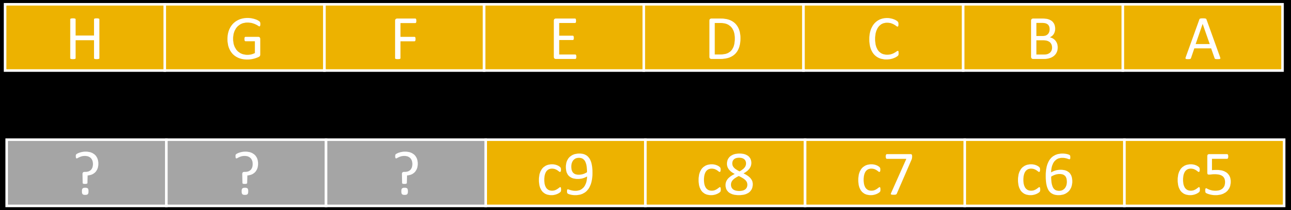
- To enable OOO execution, you need renaming
- Renaming remaps the virtual program registers to physical registers
- Physical registers are taken from a not-necessarily-sorted FREE LIST
- Example
  - `vsetvli x9 ← x8, e64, m1` ➔ `vsetvli p2 ← p28, e64, m1`
  - `vadd.vv v7 ← v5, v6` ➔ `vadd.vv vp11 ← vp2, vp17`
  - `vadd.vv v7 ← v7, v9` ➔ `vadd.vv vp29 ← vp11, vp28`
- Please note that v7 is re-mapped in the second instruction

Renaming and "Background Data"

```
vsetvli x9 ← #8, e64, m1 → vsetvli  p2   ← p28, e64, m1
vadd.vv v7 ← v5, v6      → vadd.vv  vp11 ← vp2, vp17
vsetvli x9 ← #5, e64, m1 → vsetvli  p13  ← p28, e64, m1
vadd.vv v7 ← v7, v9      → vadd.vv  vp29 ← vp11, vp28
```

1st VADD writes vp11

2nd VADD writes vp29



- IN-ORDER machine: NO worries, "H", "G" and "F" are just there
- OUT-OF-ORDER machine: "H", "G", "F" are in vp11, but not in vp29
  - A COPY IS NEEDED, or
  - VP11 must be an additional source to the second VADD
  - Specially painful if your vector is 4096 bits and your VL only asks for, say, 128b worth of work

Renaming Vector State

- A vector instruction has far more than 3 sources
- vadd.vv   v4 $\leftarrow$ v8, v12, v0.t
  - V8 in current SEW format
  - V12 in current SEW format
  - V0 in mask format
  - LMUL
  - SEW
  - VL
- In fully OOO machine, need to track them all
  - Simplifications are possible, of course

A short example

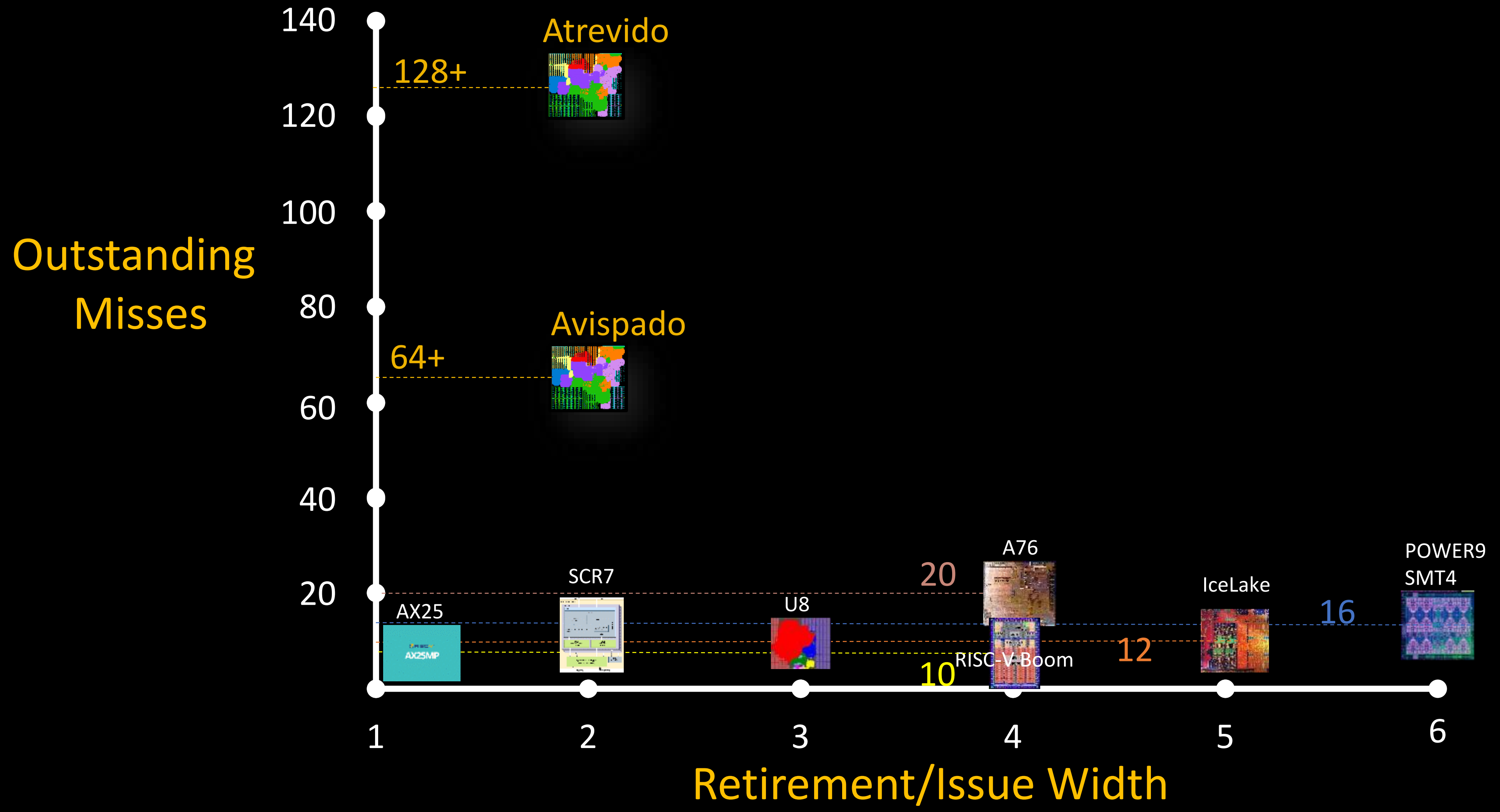
- `vsetvli x9 ← x8, e64, m1`
  - `VL = min(x8, MAXVL)`
  - `VTYPE = "e64, m1"`
  - `vsetvli <pVL5, pVTYPE3, p20> ← p12, e64, m1`
- `vadd.vv v7 ← v5, v6`
  - Rename v5 and v6
  - Rename VL
  - Rename VTYPE
  - Potentially 4 sources, optionally one more for v0, and one more for "background"
  - `vadd.vv vp20 ← vp17, vp15, pVL5, pVTYPE3`
- Think of the wakeup logic in an OOO instruction window
  - Must choose between full renaming of state, partial renaming
  - Otherwise, risk stalling the OOO on each vsetvl

Gazzillion Misses™

Definition *The ability of Semidynamics' cores to generate a very large number of outstanding memory requests*

Informally *A ton of bandwidth, Good for big data, HPC and AI*

Comparison to other cores



Gazzillion Misses™ good for…

Machine Learning

Recommendation Systems

Key-Value Stores

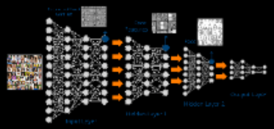
Sparse Data / HPC

Microsoft Cognitive Toolkit

MLP

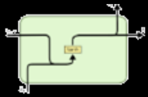
AE

CNN

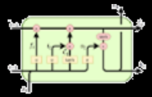


DRL

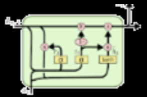
RNN



LSTM

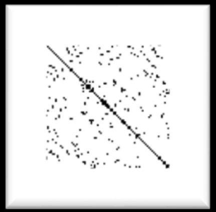


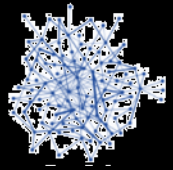
GRU

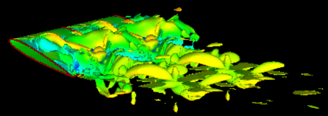


redis

Memcached

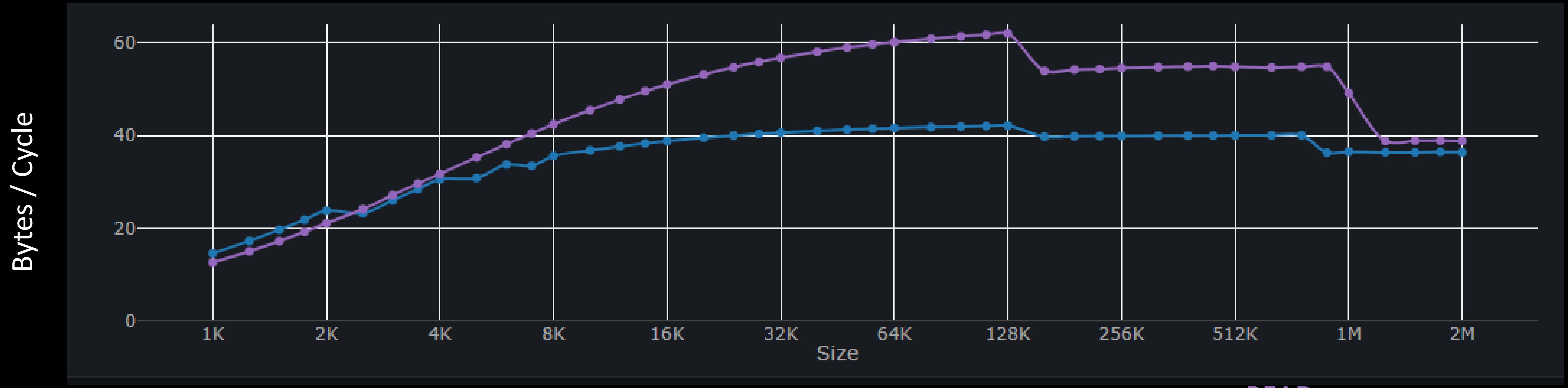






Gazzillion Misses™ incredibly good for RVV

Can you find a core out there capable of streaming data at over 60 Bytes/cycle?



READ

WRITE

Semidynamics RISC-V Cores

- Application cores, in-order and out-of-order, coherent
- Great for “Big data”, “high bandwidth” situations
- Great for RVV

- Happy to share PPA with you under NDA
- Happy to customize the core & the vpu for you

Thank you!

Thank you!